PENGGUNAAN *FUZZIPC* DAN PENGALI FREKUENSI UNTUK MENGENDALIKAN KECEPATAN MOTOR *DC*

Hany Ferdinando[(1)], Hendrik Thiehunan[(2)], Julianto Seno Putra[(3)]

Abstract: Motor speed controller usually used in industrial, so alot of researches done to make better speed control system. In this paper the used of FuzzIPC (Driver Fuzzy logic for Festo) in controlling DC motor speed. As controller using Festo PLC FC34 type. Constraint of Festo PLC gives Pulse Width Modulation (PWM) that must use external frequency multiplier built using AT89C2051 WITH 12MHz that change PWM signal from PLC to give new PWM signal with higher frequency, but has the same duty cycle. Experiment result shows that frequency multiplier modul output has cbigger multipier factor cycle duty error. The biggest error occur multiplier factor 250. FuzzyIPC used as controller algorithm can give a good control DC motor speed. Frequency multiplier must be revised still has big error. Non-accuracy system in calculating and speed of calculation done by microcontroller.

Keywords: FuzzIPC, Frekuency Multiplier, Fuzzy, PLC, Festo

Pengendalian motor merupakan salah satu *plant* yang umum dipergunakan. Selain karena banyak diaplikasikan di industri, *plant* ini mudah dibuat. Hal ini memicu berbagai macam pemikiran untuk terus mengembangkan algoritma yang makin baik. Salah satu algoritma yang masih terus dikembangkan adalah *Fuzzy Logic*.

Untuk melakukan pengendalian berbasis *Fuzzy Logic*, diperlukan suatu perhitungan yang rumit. *PLC* sebagai salah satu pengendali yang handal di industri biasanya belum dilengkapi dengan *Fuzzy Logic*, terutama untuk *PLC* tipe lama. Oleh karena itu, dikembangkan suatu *driver Fuzzy Logic* untuk *PLC*. Salah satu *PLC* yang dipilih adalah FESTO (Setiawan, 2003).

PLC FESTO memiliki keterbatasan dalam menghasilkan sinyal *PWM*. Sinyal *PWM* dengan frekuensi 20Hz (terendah) dapat menggunakan perubahan dutu *cycle* per 1%. Sedangkan, *PWM* dengan frekuensi 1kHz (tertinggi) hanya memiliki tiga pilihan, yaitu 0%, 50% dan 100%.

Penggunaan sinyal *PWM* berfrekuensi rendah akan menyebabkan putaran motor tersendat-sendat. Hal ini mengakibatkan pengendalian motor tidak dapat dilakukan dengan baik. Untuk itu, diperlukan suatu modul pengali frekuensi eksternal yang akan menaikkan frekuensi sinyal *PWM input* tanpa mengubah *duty cycle*.

Makalah ini, membahas pembuatan modul pengali frekuensi yang akan digabungkan dengan

(1) Hany Ferdinando, Jurusan Teknik Elektro, Universitas Kristen Petra, Surabaya
(2) Hendrik Thiehunan, Jurusan Teknik Elektro, Universitas Kristen Petra, Surabaya
(3) Julianto Seno Putra, Jurusan Teknik Elektro, Universitas Kristen Petra, Surabaya

PLC FESTO untuk mengendalikan kecepatan motor *DC*. Algoritma yang dipergunakan adalah *Fuzzy Logic*.

Fuzzy Logic

Keberadaan *Fuzzy Logic* sudah merupakan bagian yang tak terpisahkan. Teori yang berawal dari konsep himpunan dan konsep logika yang tidak '0' dan '1' ini telah berkembang dengan sangat pesat.

Fuzzy menggunakan konsep himpunan dan *input* yang ada diubah menjadi derajad keanggotaan menurut tiap himpunan. Pengambilan keputusan dilakukan dengan menggunakan suatu aturan yang berbasis *if – then*. Hasil dari pengolahan menggunakan aturan ini masih dalam *Fuzzy*, sehingga perlu diubah ke dalam variabel yang dapat dimengerti dunia kita.

Secara konseptual, *Fuzzy* melakukan proses *Fuzzyfication* (mengubah variabel di dunia kita menjadi variabel di dunia *Fuzzy*), proses pengambilan keputusan menggunakan aturan dan proses penggabungan berbagai macam hasil dari aturan yang dipergunakan untuk diterjemahkan ke dalam dunia kita). Perhitungan yang dilakukan untuk mendapatkan hasil akhir cukup panjang. Sehingga akan sangat bermanfaat jika dapat dibuat sebuah *driver* yang berbasis *Fuzzy Logic*.

FuzzIPC

FuzzIPC adalah *driver* yang dikembangkan untuk *PLC FESTO* sehingga dapat menggunakan *Fuzzy Logic* sebagai algoritma pengendaliannya (Setiawan, 2003). *FuzzIPC* dikembangkan dari *PetraFuz* yang membuat proses perancangan dan implementasi algoritma *Fuzzy* menjadi lebih mudah dan cepat. *FuzzIP* telah dicoba untuk mengendalikan suhu pada sebuah ruangan yang dipergunakan untuk menguji ketahanan sebuah alat (Ferdinanto, 2005).

Pulse Width Modulation

Pulse Width Modulation atau lebih dikenal dengan *PWM* merupakan metode untuk mendapatkan sinyal *DC* yang bervariasi yang dikendalikan secara digital. Sinyal ini merupakan gelombang kotak dengan frekuensi tetap dengan variasi pada lebar pulsa *HIGH*. Rasio lebar pulsa (dalam satuan waktu) terhadap periodenya disebut dengan *duty cycle*. *Duty cycle* 10% artinya lebar pulsa *HIGH* adalah 10% dari periode sinyal kotak.

Variasi *duty cycle* ini memberikan harga tegangan rata-rata yang berbeda-beda. Sinyal *PWM* dengan *duty cycle* yang besar memiliki harga rata-rata yang lebih besar dibandingkan yang memiliki *ducy cycle* kecil. Nilai tegangan yang diberikan sebanding dengan nilai *duty cycle* yang diberikan.

Penggunaan sinyal *PWM* untuk mengendalikan kecepatan motor *DC* sudah umum dilakukan. Salah satu faktor penting dalam penggunaan sinyal ini adalah besarnya frekuensi yang dipilih. Membangkitkan sinyal dengan frekuensi rendah lebih mudah dibandingkan frekuensi tinggi. Tetapi sinyal *PWM* dengan frekuensi rendah tidak memberikan hasil yang memuaskan jika diaplikasikan pada motor *DC*. Putaran motor *DC* pada sinyal *PWM* frekuensi rendah akan tersendat-sendat.

PWM* Pada *PLC FESTO

PLC FESTO dapat membangkitkan sinyal *PWM* dengan rentang frekuensi 20Hz sampai dengan 1kHz. Akan tetapi terdapat suatu karakteristik yang membatasi penggunaan sinyak *PWM* pada *PLC*

FESTO.

Sinyal *PWM* dengan frekuensi 20Hz dapat diatur *duty cycle* per 1%. Hal ini berbeda dengan sinyal dengan frekuensi 1kHz. Sinyal ini hanya memiliki 3 kemungkinan *duty cycle*, yaitu 0%, 50% dan 100%. Sinyal pada frekuensi 200Hz memiliki resolusi 10% (Festo, 1999).

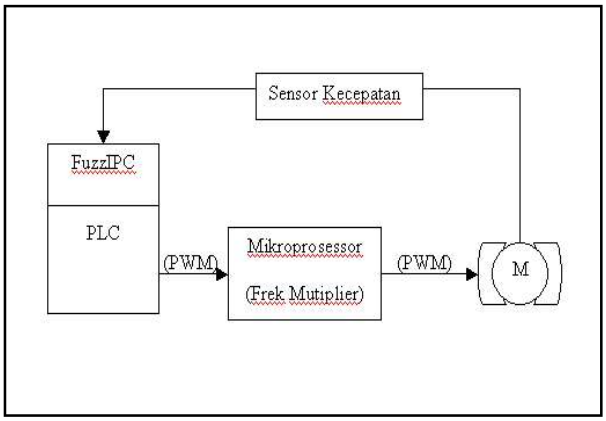
Dalam berbagai macam aplikasi diperlukan pengaturan *duty cycle* per 1%. Namun, jika sinyal ini diambil dari *PLC FESTO*, maka sinyal yang dipergunakan harus 20Hz. Sinyal *PWM* dengan frekuensi rendah akan menyebabkan putaran motor tersendat. Hal ini akan menyebabkan pengendalian motor *DC* terganggu. Salah satu solusi yang dapat dipakai adalah menggunakan pengali frekuensi untuk menaikkan frekuensi sinyal *PWM* tetapi tidak mengubah *duty cycle*.

Pengali Frekuensi

Pengali frekuensi yang dibahas dalam makalah ini diaplikasikan untuk sinyal kotak. Sistem membaca lebar pulsa *HIGH* dan *LOW* untuk mendapatkan informasi sinyal masukan. Apabila pengali yang dipilih adalah dua, maka informasi ini akan dibagi dengan faktor pengali yang dipergunakan. Sebagai contoh sinyal *PWM* dengan frekuensi 20Hz dan *duty cycle* 50% memiliki periode 50ms. Sehingga lebar pulsa *HIGH* adalah 25ms, demikian juga dengan lebar pulsa *LOW*. Jika diinginkan sinyal *PWM* dengan frekuensi 100Hz, maka faktor pengali yang dipergunakan adalah 5. Sehingga pulsa *HIGH* selebar 25ms dibagi dengan 5, menghasilkan pulsa *HIGH* dengan lebar 5ms. Hal sama dilakukan dengan pulsa *LOW*. Periode sinyal yang baru menjadi 10ms. Ini setara dengan 100Hz.

Desain Perangkat Keras

Sistem pengali frekuensi ini akan diimplementasikan pada pengendalian kecepatan motor *DC*. Algoritma yang dipergunakan adalah *Fuzzy Logic*. Gambar 1 menunjukkan *block* diagram dari sistem yang dibuat.



Gambar 1 *Block* Diagram Sistem

Sistem dibuat dengan pilihan faktor pengali 1 s/d 255. Hal ini ditentukan berdasarkan posisi dip *switch* yang terhubung dengan *microcontroller* AT89C2051. Algoritma *Fuzzy* diletakkan pada *PLC FESTO* FC34. Motor yang dipergunakan memiliki tegangan kerja 12V dan dilengkapi dengan internal *encoder*. Sistem menerima *input* sinyal *PWM* dengan frekuensi 20Hz, karena pada frekuensi inilah *duty cycle* dapat diset dengan resolusi 1%.

Untuk mengukur sinyal dengan baik sistem menerima sinyal masukan menggunakan pin *interrupt* eksternal. Karena pin diset untuk membaca transisi dari *HIGH* ke *LOW*, maka diperlukan sebuah rangkaian inverter untuk mendapatkan hasil yang diinginkan. Sinyal dari *PLC* menggunakan *level* 0-24VDC, sedangkan AT89C2051 menggunakan *level* 0-5VDC. Oleh karena itu dipergunakan sebuah pembagi tegangan untuk menurunkan *level* dari *PLC*.

Motor yang dipergunakan sudah dilengkapi dengan *encoder*. Kecepatan putaran motor diukur berdasarkan jumlah pulsa yang diterima oleh PLC dalam waktu 1ms. Semakin cepat putaran motor, maka semakin banyak jumlah pulsa yang dibaca, dan sebaliknya. Namun terdapat perbedaan *level* tegangan logika untuk *encoder* dan *PLC*. Sebagaimana diketahui, *encoder* menggunakan *level* 0-5V sedang PLC menggunakan 0-24V. Untuk mengatasi hal ini digunakan sebuah *inverter* dengan menggunakan sebuah *optocoupler*.

METODE

Desain Perangkat Lunak

AT89C2051 memiliki 2 buah *timer* yang difungsikan untuk membaca lebar pulsa *HIGH* dan *LOW* sinyal *input* (Timer0) dan membangkitkan sinyal PWM yang baru (Timer1). Dengan kristal 12 MHz, maka dalam 1 *cycle* memerlukan waktu 1 μ s. Nilai ini akan menjadi dasar pada perhitungan yang dilakukan nanti (Christanto dan Pusporini, 2004).

Pada saat terjadi transisi *HIGH* ke *LOW*, pin *interrupt* akan mendeteksinya. Informasi ini dipergunakan untuk melakukan *start timer* yang dipergunakan (ini untuk menghitung pulsa *HIGH*). Pada saat transisi dari *LOW* ke *HIGH*, maka pulsa *HIGH* selesai dihitung. Karena dimasukkan ke sebuah rangkaian *NOT*, maka transisi ini akan menjadi *HIGH* ke *LOW*. Sinyal ini akan menghentikan *timer* yang di-*start* tadi. Nilai *timer* yang tersimpan adalah untuk lebar pulsa *HIGH*. Setelah nilai *timer* ini disimpan di lokasi tertentu, maka *timer* di-*start* lagi untuk menghitung pulsa *LOW*. Pada saat ada transiri dari *LOW* ke *HIGH*, rangkaian *NOT* akan berfungsi kembali dan proses yang sama akan terulang.

Nilai *timer* yang dibaca tadi tersimpan dalam *register* TH0 dan TL0. Nilai TH0 dan TL0 ini membentuk data 16-bit dan akan mengalami pembagian sesuai dengan faktor pengali yang dipilih.

Berikut adalah algoritma pembagian TH dan TL untuk mendapatkan sinyal *PWM* yang baru. Frekuensi sinyal yang masuk ditetapkan 20Hz atau 50ms. Dengan *duty cycle* 50%, maka $T_H = T_L = 25$ms. Persamaan (1) menunjukkan perhitungan TH0 TL0.

$$TH_0TL_0 = \frac{25ms}{1\mu s} = 25000 \quad (1)$$

Nilai 25000 ini terdistribusi dalam THx dan TLx. Nilai THx adalah

$$TH_0 = \frac{25000}{256} \cong 97 \quad (2)$$

Nilai ini adalah pembulatan ke bawah. Nilai TLx sedikit lebih kompleks, yaitu

$$TL_0 = 25000 - 256 * 97 = 168 \quad (3)$$

Nilai THx akan bertambah 1 apabila TLx mengalami *overflow*, artinya saat terjadi transisi dari 255 ke 0. Hal ini terjadi pada *Timer0* maupun *Timer1*. Sehingga apabila TH0 sudah mengalami 97 kali *overflow* dan TL0 saat itu bernilai 168, maka nilai *timer* yang dibaca setara dengan 25000 atau 25ms.

Dengan faktor pengali 50, maka nilai 25000 ini dibagi dengan 50, hasil yang diharapkan adalah 500. Berikut cara mendapatkan nilai ini untuk TH0 dan TL0 yang baru

$$TH_0^{'} = \frac{TH_0}{50} = \frac{97}{50} = 1 \quad (4)$$

Perhitungan ini masih menyisakan nilai 47 sebagai sisa hasil bagi. Karena THx adalah 256 kali dari TLx, maka sisa ini tidak dapat diabaikan. Lalu dilakukan perhitungan terhadap TLx. Pembagian untuk TLx juga menghasilkan sisa.

$$TL_0^{'} = \left(sisa\_TH_0 \frac{255}{50} \right) + \frac{TL_0}{50}$$
$$TL_0^{'} = 47 \frac{255}{50} + \frac{168}{50} \quad (5)$$

Dalam perhitungan, 255 dibagi 50 akan menyisakan 5, sehingga akan terdapat akumulasi sisa sebanyak 47 kali 5. Nilai masih memberikan kontribusi pada TL0 yang baru (setelah dibagi). Nilai ini sebesar

$$sisa = \frac{47*5}{50} \cong 4 \quad (6)$$

Masih terdapat sisa sebesar 35. Nilai ini diabaikan terhadap faktor pengali yang ada. Sehingga TLx yang baru setelah dilakukan berbagai macam penambahan menjadi

$$TL_0^{'} = 3 + 235 + 4 = 242 \quad (7)$$

Hasil akhir TH0 dan TL0 yang baru masing-masing adalah 1 dan 242. Nilai ini memberikan nilai timer 1 kali 256 ditambah 242, yaitu 498. Terdapat error 2 satuan terhadap nilai yang seharusnya dipergunakan.

Perhitungan ini dilakukan untuk T_H dan T_L pada sinyal *PWM*. Karena dalam contoh ini dipergunakan *duty cycle* 50%, maka perhitungan yang sama akan dilakukan dua kali. T_H yang baru menjadi

$$T_H = 498*1\mu s \quad (8)$$

T_L juga memiliki nilai yang sama. Frekuensi sinyal *PWM* yang baru menjadi

$$f = \frac{1}{(498+498)10^{-6}} Hz = 1004Hz \quad (9)$$

Nilai yang diharapkan sebesar 1000Hz. Kesalahan ini terjadi akibat pembulatan yang dilakukan pada saat proses pembagian berlangsung.

Hasil TH0 dan TL0 yang baru langsung dimasukkan ke register TH1 dan TL1 yang berfungsi untuk membangkitkan sinyal *PWM* dengan frekuensi yang diinginkan dan *duty cycle* sama dengan sinyal *PWM input*.

PLC sebagai pengendali menerima data kecepatan putaran motor dengan menggunakan *encoder*. Percobaan awal yang dilakukan terhadap motor menunjukkan bahwa kecepatan maksimum motor adalah 3600 *rpm* (*rotation per minute*) dengan jumlah pulsa berkisar 7000 pulsa per detik. Tetapi, kemampulan *fast counter PLC FESTO* 2kHz. Sehingga pulsa ini harus dibagi 4 agar dapat dibaca oleh *PLC*. Sehingga, jumlah pulsa yang dihitung oleh *PLC* maksimum 1750 pulsa per detik dan ini setara dengan 3600 *rpm*.

Fuzzy menggunakan dua *input*, yaitu *error* dan *derror*. *Error* adalah selisih antara kecepatan sekarang (*present value-PV*) dengan kecepatan referensi (*setting point-SP*). *Derror* adalah selisih antara *error* sekarang dengan *error* sebelumnya. Hal ini memberikan informasi arah kecenderungan *system*. Apabila *PV* sudah sama dengan *SP*, maka nilai *error* akan nol. Tetapi kecenderungan pergerakan *PV* ini tidak diketahui sampai perhitungan derror didapatkan.

Program pada *PLC FESTO* ditulis dalam bahasa *statement list* (Festo, 1999). *Listing* 1 adalah penggalan program yang dibuat:

***Listing* 1 *Program PLC* FESTO**

```
STEP 0
   IF   N     T_S      'TimeSampling
   THEN LOAD  PV 'PV  (Rps)
        -     SP 'SP  (Rps)
        TO    ERROR 'Error  (1)

        LOAD L_ERROR  'Error  (n-1)
        - ERROR 'Error  (1)
        TO dERROR  'Selisih  Error

        LOAD ERROR 'Error  (1)
        TO L_ERROR 'Error  (n-1)

STEP NormFuzz
   THEN LOAD ( ERROR  'Error  (1)
```

```
        + V64 )
        * V2
        TO N_ERROR 'normalisasi

        LOAD ( dERROR 'Selisih Error
        + V64 )
        * V2
        TO N_dERROR'normalisasi

STEP FuzDefuzz
   THEN CFM 2 'TA30.COM
        WITH N_ERROR
        WITH N_dERROR
        LOAD FU32
        TO FuzzOUT 'Output
        LOAD PV
        TO R100
        LOAD SP
        TO R101

STEP NormDefuz
""NormOUTPUT AND OnTime Reload (0-128 = 64-
0)
   IF   ( FuzzOUT
          <= V128 )
   THEN  LOAD V64
         - ( ( FuzzOUT
         * V64 )
         / V128 )
         TO  F_OUT 'normalisasi

         LOAD ON 'Duty cycle (%)
         - F_OUT
         TO  ON

""NormOUTPUT AND OnTime Reload (128-255 =
0-64)
   IF   ( FuzzOUT
        > V128 )

   THEN LOAD ( ( FuzzOUT
        * V64 )
        / V128 )
        - V64
        TO F_OUT 'normalisasi
        LOAD ON 'Duty Cycle (%)
        + F_OUT
        TO ON
   :
   :
        JMP TO 0
```

PLC menggunakan modul bernama *fastout* untuk membangkitkan sinyal *PWM*. Frekuensi yang dipilih 20Hz karena pada frekuensi inilah pengguna dapat mengatur perubahan *duty cycle* per 1%.

Dalam menggunakan modul ini, ada beberapa *function unit* yang harus diatur. *Function unit* atau yang biasa disebut dengan *FU* ini berfungsi untuk melewatkan informasi dari dan ke pengguna program dan modul fungsi [3]. Pada modul *fastout* ini terdapat 5 *FU input* yang terdiri dari *FU*32 (0: *init*; 1: *stop output*; 2,3,4,5: *start*), *FU*33 (*Output number* 0….7), *FU*34 (*On time* *0.5mS), *FU*35 (*Off time* *0.5mS), *FU*36 (*Number of pulses*) dan 3 *FU output* yang terdiri dari *FU* 32 (0 *if successful*; 100 *if driver not installed*), *FU*33 (1 *if output started and not yet finished*), *FU*34 (*Number of remaining pulse*). *Listing* 2 adalah penggalan program pembangkit pulsa:

***Listing* 2 Program Pembangkit Pulsa**

```
STEP init
   THEN CFM 1          'FASTOUT
   WITH V0             "Init Fast Out
   :
   :

STEP Run
   THEN LOAD V100     "Max Pulse (%)
        -   ON        'DutyCycle ON (%)
        TO  OFF       'DutyCycle OFF (%)
   :
   :

   IF   T_S           'TimeSampling
   AND  F10.0         'Start Flag
   THEN CFM 1         'FASTOUT
        WITHV5        "Start
        WITHV2        "Output Number
        WITHON        'DutyCycle ON (%)
        WITHOFF       'DutyCycle OFF (%)
        WITHV5        "Number of Pulse
        RESET         F10.0 'Start Flag

   IF   F10.2         'Stop Flag
   THEN CFM 1         'FASTOUT
        WITH V1       "Stop
        WITH V2
        RESET F10.2   'Stop Flag
   :
   :
   JMP TO Run
```

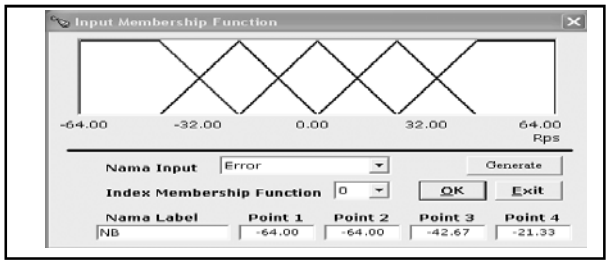
Bagian program berikut yang dibuat adalah pembaca pulsa untuk menghitung kecepatan motor. Program ini menggunakan *fast counter* [3], yang menghitung jumlah pulsa dalam 1 detik. Pada modul ini terdapat beberapa *function unit* yang dipakai, yaitu FU32 (0: *reset*; 1: *parameterizing*; 2: *activating*; 3: *interrogating the status and current counter*

value), FU33 (0: *first counter*; 1: *second counter*). *Listing* 3 menunjukkan penggalan program pembaca pulsa.

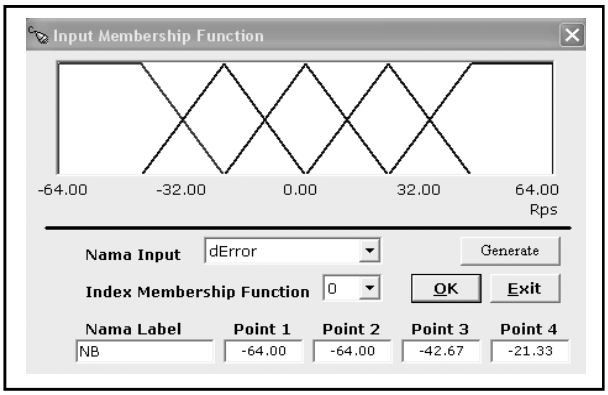
***Listing* 2 Program Pembaca Pulsa**

```
STEP 0
   THEN LOAD V50
        TO     TP_1s   'Timer Preset 1s
        SET    T_1s    'Timer 1s

    CFM 0              'FECCNTR
          WITHV0       "Reset Counter
          WITH  V0     "First Counter

   IF   NOP
   THEN  CFM 0         'FECCNTR
          WITHV2       "Start Counter
          WITH  V0     "First Counter

STEP 1
   IF   N     T_1s     'Timer 1s
   THEN LOAD   PULSE   'OutputEncoder(Hz)
          /     V15
          TO    PV     'PV (Rps)

   CFM 0               'FECCNTR
          WITHV0       "Reset Counter
          WITH  V0     "First Counter
   CFM 0 'FECCNTR
          WITHV2       "Start Counter
          WITH  V0     "First Counter
        SET  T_1s      'Timer 1s
   :
   :

   JMP TO 1
```

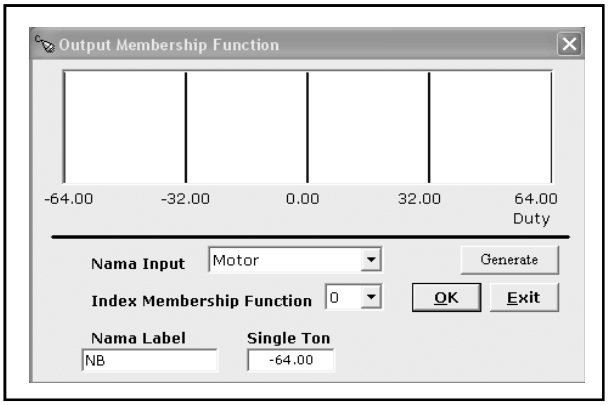
Gambar 5 menunjukkan bentuk *membership function* untuk *input error* dan *derror,* sedangkan Gambar 6 menunjukkan bentuk *membership function* untuk *output*. FuzzIPC hanya menerima *output* dalam bentuk *singleton*. Hal ini dilakukan untuk mempermudah perhitungan akhir. Aturan yang dibuat dapat dilihat pada Gambar 7.



Gambar 5a Bentuk *Membership Function Input: Error*



Gambar 5b Bentuk *Membership Function Input: Derror*



Gambar 6 Bentuk *Membership Function Output*

		Error				
		NB	NS	Z	PS	PB
Derror	NB	PS	NS	NB	NB	NB
	NS	PS	PS	NS	NS	NB
	Z	PB	PS	Z	NS	NB
	PS	PB	PS	PS	NS	NS
	PB	PB	PB	PB	PS	NS

Gambar 7 Rancangan Aturan pada *Fuzzy*

HASIL PEMBAHASAN

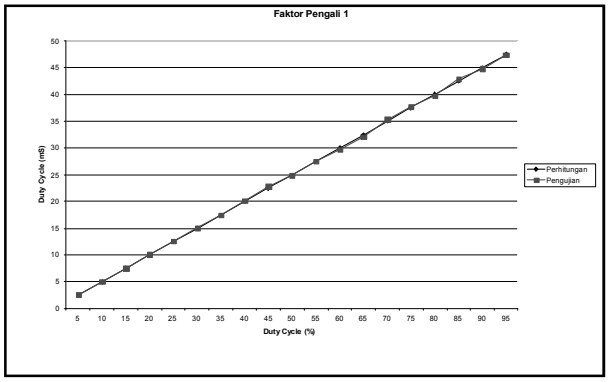
Pengujian awal dilakukan dengan menggunakan *duty cycle* 50% untuk beberapa variasi faktor pengali. Tabel 1 menunjukkan adanya perbedaan antara perhitungan dengan hasil pengukuran. Hal ini terjadi oleh karena pembulatan yang sudah dijelaskan pada bagian sebelum ini.

Tabel 1 Hasil Pengujian Awal

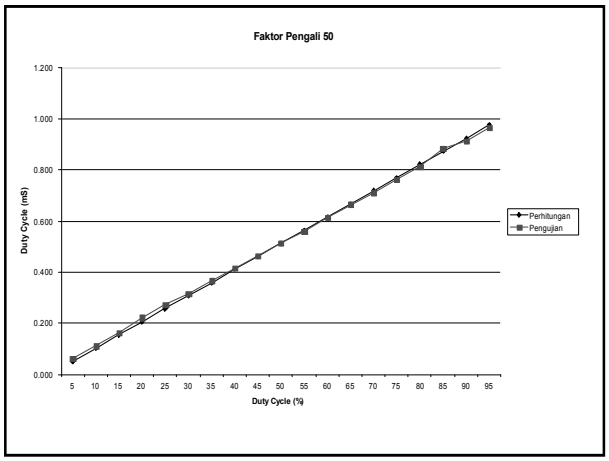
Faktor Pengali	Perhitungan (Hz)	Pengujian (Hz)	Selisih (Hz)
1	20	20.0	0.0
10	200	200.0	0.0
20	400	394.5	-5.5
30	600	588.2	-11.8
40	800	772.2	-27.8
50	1000	972.8	-27.2
60	1200	1157.4	-42.6
70	1400	1362.4	-37.6
80	1600	1538.5	-61.5
90	1800	1712.3	-87.7
100	2000	1912.0	-88.0
110	2200	2083.3	-116.7
120	2400	2252.3	-147.7
130	2600	2500.0	-100.0
140	2800	2611.0	-189.0
150	3000	2809.0	-191.0
160	3200	2941.2	-258.8
170	3400	3125.0	-275.0
180	3600	3300.3	-299.7
190	3800	3484.3	-315.7

Sumber: Hasil Percobaan

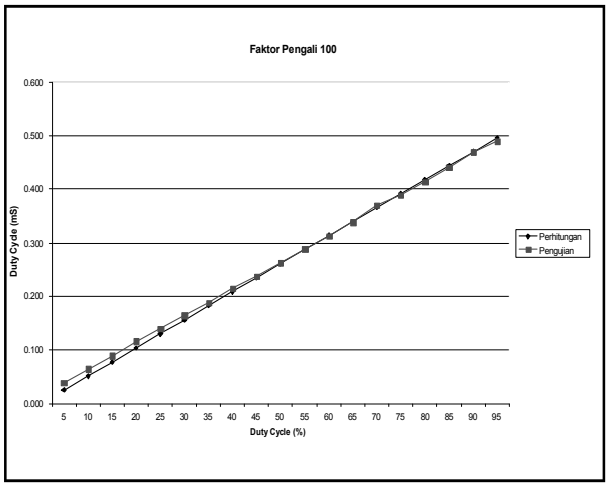
Gambar 8 sampai dengan Gambar 15 menunjukkan hasil pengujian untuk variasi *duty cycle* 5-95% dengan berbagai faktor pengali. Lambang kotak merupakan hasil pengukuran sedang lambing *diamond* merupakan hasil perhitungan



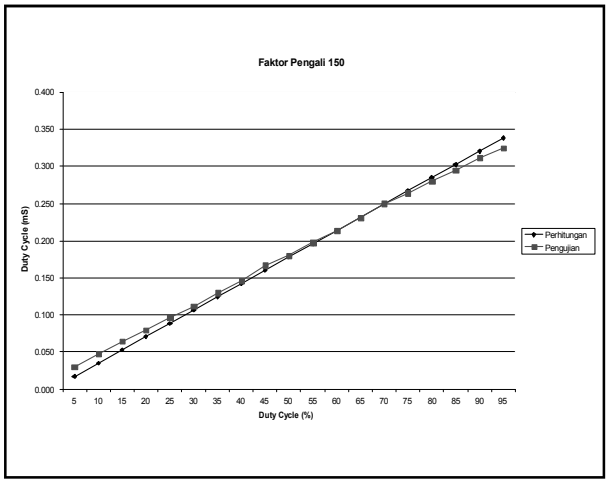
Gambar 8 Faktor Pengali 1



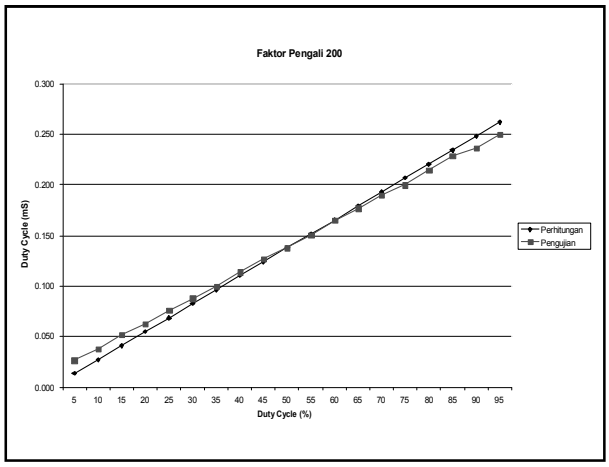
Gambar 9 Faktor Pengali 50



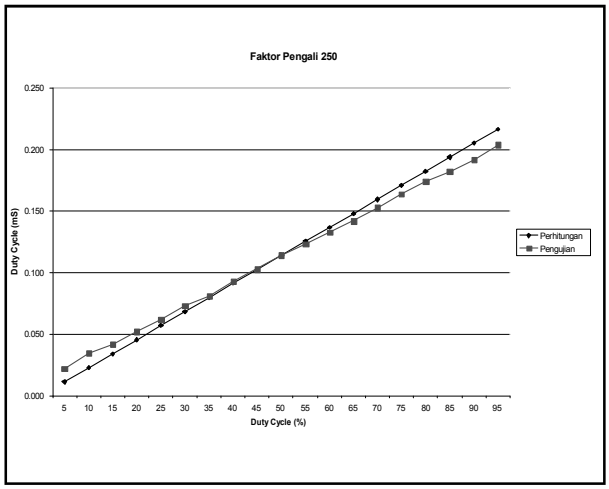
Gambar 10 Faktor Pengali 100



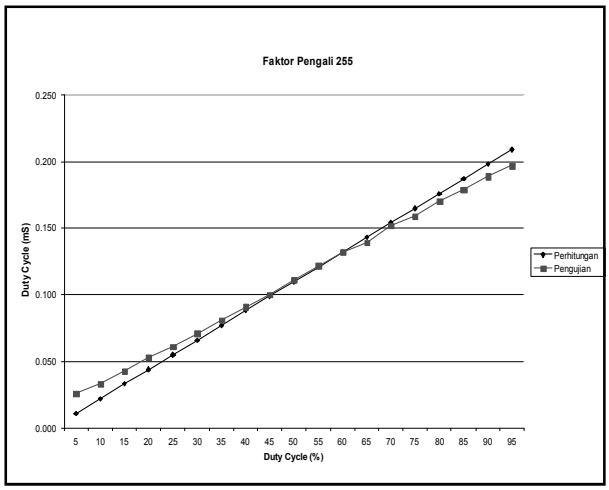
Gambar 11 Faktor Pengali 150



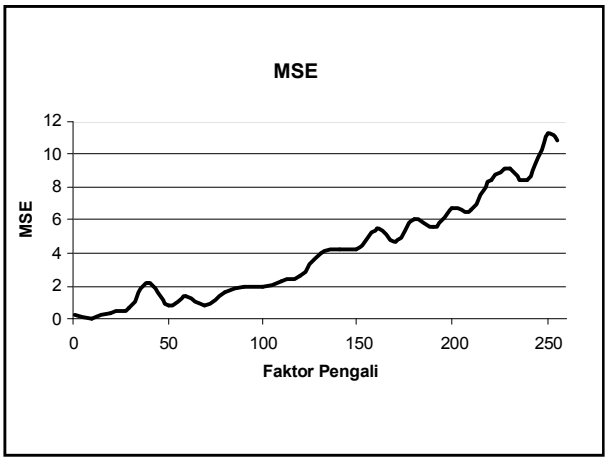
Gambar 12 Faktor Pengali 200



Gambar 13 Faktor Pengali 250



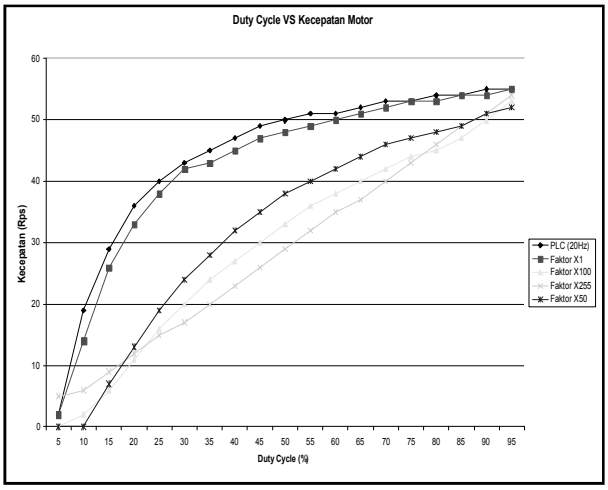
Gambar 14 Faktor Pengali 255



Gambar 15 Ringkasan Pengujian untuk Semua Faktor Pengali

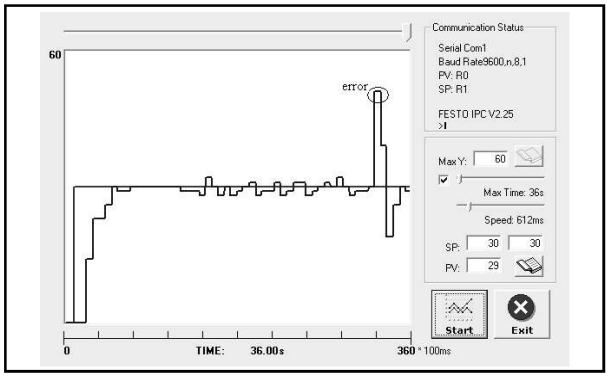
Dari semua pengujian faktor pengali ini didapatkan bahwa *error system* semakin besar seiring dengan semakin besarnya faktor pengali. Hal ini dikarenakan pada faktor pengali yang tinggi, sinyal *PWM* yang dihasilkan memiliki frekuensi yang lebih tinggi. Sinyal frekuensi tinggi memerlukan waktu yang lebih cepat dibandingkan dengan frekuensi rendah.

Pengujian berikutnya adalah penggabungan modul pengali frekuensi ini dengan program *fuzzy* di dalam *PLC*. Gambar 16 menunjukkan ringkasan percobaan respon motor yang dilakukan pada beberapa faktor pengali.

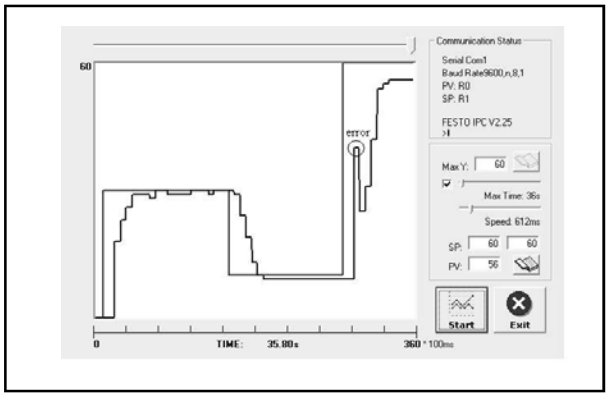


Gambar 16 Respon Motor dengan Berbagai Faktor Pengali

Gambar 16 menunjukkan hasil yang menarik, yaitu respon motor dengan dan tanpa modul pengali frekuensi hampir sama. Terlihat juga bahwa pada faktor pengali yang kecil (*PWM* dengan frekuensi rendah), respon motor tidak linier terhadap perubahan *duty cycle.*



Gambar 17 Pengujian Respon Motor Menggunakan *Fuzzy Logic*



Gambar 18 Respon Sistem untuk Mengatasi Gangguan

Respon sistem terlihat cukup baik dengan *rise time* berkisar 0,6 detik. Terlihat bahwa terdapat lonjakan pada akhir pengujian, saat sistem sudah stabil. Pada beberapa pengujian yang lain, lonjakan seperti itu masih terlihat dan polanya acak.

Gambar 18 menunjukkan respon sistem pada gangguan yang diberikan. Terlihat bahwa sistem dapat mengatasi gangguan yang terjadi.

SIMPULAN

Hasil pembagian sinyal *PWM* dengan frekuensi 20Hz tidak berjalan dengan baik. Performansi sistem yang ditunjukkan dengan nilai *MSE* dari *duty cycle*. Nilai *MSE* ini secara umum berbanding lurus dengan faktor pengali yang dipilih. Pada faktor pengali 50, seperti contoh di atas mengabaikan sisa hasil bagi sebesar 35. Untuk faktor pengali yang lebih besar, misalnya 200, sangat mungkin terjadi pembuangan sisa hasil bagi yang nilanya mendekati 200. Pada contoh di atas, nilai yang diabaikan adalah 35, ini berarti terdapat kesalahan sebesar 35 μ s. Apabila untuk faktor pengali 200, sisa hasil bagi yang diabaikan adalah 199, terdapat kesalahan sebesar 199 μ s atau 0.199ms. Di sini terlihat bahwa kesalahan yang mungkin terjadi akan semakin besar untuk faktor pengali yang semakin besar juga.

Pada beberapa percobaan, terdapat suatu kejadian saat motor hilang kendali beberapa saat. Hal ini sangat mungkin terjadi karena *microcontroller* harus membaca lebar pulsa dan membangkitkan sinyal *PWM*. Karena keduanya memanfaatkan fasilitas *interrupt* dan hanya satu yang dapat dilayani, maka penangannya terlihat tidak berjalan sebagaimana mestinya. Sistem memang diatur untuk memprioritaskan pembacaan lebar pulsa. Penggunaan frekuensi *clock* yang lebih besar diharapkan dapat mengatasi hal ini.

Error yang terjadi akibat ketidakmampuan *microcontroller* melakukan perhitungan lebar pulsa dan pembangkitan sinyal *PWM* sedikit teratasi dengan digunakannya logika *fuzzy* pada pengendalian *plant*. Hal ini terlihat dari Gambar 17 saat sistem dapat segera kembali ke titik stabil dengan cepat. Respon terhadap gangguan yang diberikan (Gambar 18) menunjukkan bahwa *fuzzy logic* dapat mengatasinya

dengan baik. *Tuning* yang dilakukan pada sistem tidak terlalu banyak berpengaruh untuk memperbaiki *respon* sistem.

RUJUKAN

Christanto, D, dan Pusporini, C. 2004. *Panduan Dasar Mikrokontroller Keluarga MCS51*. Surabaya: Innovative Electronics.

Ferdinando, H., Thiehunan, H, and Basuki, H. 2005. *Fuzzle PC in Proc. of Intelligent Technology 2006*. Thailand: Assumption University.

FESTO Didactic, 1999, *The CPU I/O Module FEC FC34-FST*, Germany:_____.

Setiawan, E, 2003, *Pembuatan Modul Fuzzy Controller Untuk PLC Festo*, Tugas Akhir S-1, Surabaya: Universitas Kristen Petra